PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-063508
(43)Date of publication of application : 06.03.1998

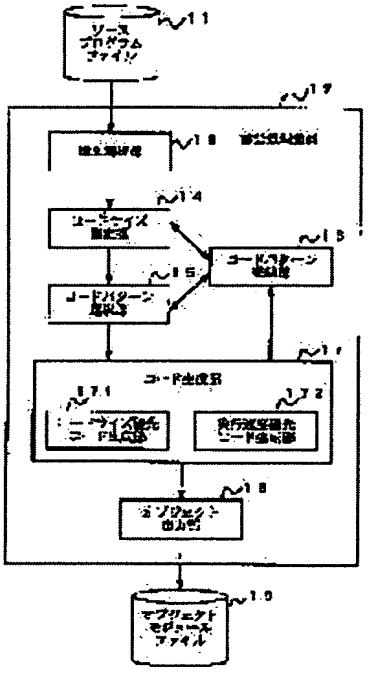
(51)Int.CI. G06F 9/45

(21)Application number : 08-237211
(22)Date of filing : 19.08.1996

(71)Applicant : NEC CORP
(72)Inventor : EGASHIRA AKIRA

(54) LANGUAGE PROCESSOR AND ITS METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a language processor and its processing method generating the object of a high executing speed within the limit of a prescribed code size.
SOLUTION: The language processor with a syntax analyzing part 13, a code generating part 17 and an object output part 18 is provided with a code pattern registering part 16 for registering prescribed information concerning a pattern outputting different codes depending on a generation code giving priority to the down-sizing of an object and a generation code giving priority to the acceleration of the execution of the object, a code size measuring part 14 measuring the size of the object and the number of the times of using each pattern of outputting codes registered in the part 16, and a code pattern selecting part 15 replacing a part and all of the generation codes giving priority to the execution speed of the object with the generation codes giving priority to the size of the object to be not larger than a prescribed code size.



LEGAL STATUS

[Date of request for examination] 19.08.1996
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3237693
[Date of registration] 05.10.2001
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－63508

(43)公開日　平成10年(1998)３月６日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/45			Ｇ０６Ｆ　9/44	３２２Ｆ	

審査請求　有　　請求項の数６　ＦＤ　（全 13 頁）

(21)出願番号　特願平8－237211

(22)出願日　平成８年(1996)８月19日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　江頭　朗
東京都港区芝五丁目７番１号　日本電気株式会社内

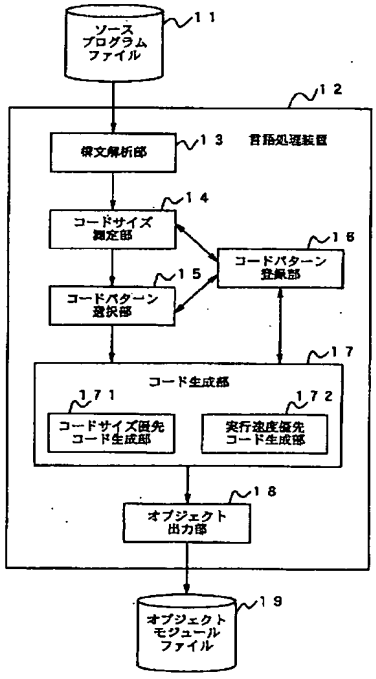
(74)代理人　弁理士　松本　正夫

(54)【発明の名称】　言語処理装置および言語処理方法

(57)【要約】

【課題】　所定のコードサイズの範囲内で実行速度の速いオブジェクトを生成することができる言語処理装置および言語処理方法を提供する。

【解決手段】　構文解析部１３と、コード生成部１７と、オブジェクト出力部１８とを備える言語処理装置において、オブジェクトのサイズを小さくすることを優先した生成コードとオブジェクトの実行速度を速くすることを優先した生成コードとで異なるコードを出力するパターンについて所定の情報を登録するコードパターン登録部１６と、オブジェクトのサイズとコードパターン登録部１６に登録された出力コードの各パターンを使用した回数とを測定するコードサイズ測定部１４と、所定のコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部１５とを備える。



1

【特許請求の範囲】

【請求項１】　ソースプログラムを入力して解析する構文解析部と、解析された結果に基づいて命令コードを生成するコード生成部と、生成された命令コードをオブジェクト・モジュール・ファイルに出力するオブジェクト出力部とを備える言語処理装置において、

オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておくコードパターン登録部と、

オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、前記コードパターン登録部に登録されている出力コードの各パターンを使用した回数を測定するコードサイズ測定部と、

前記コードサイズ測定部の測定結果と前記コードパターン登録部の登録内容を参照し、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部とを備えることを特徴とする言語処理装置。

【請求項２】　前記コードパターン登録部が、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録することを特徴とする請求項１に記載の言語処理装置。

【請求項３】　前記コードパターン選択部が、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定することを特徴とする請求項１または請求項２に記載の言語処理装置。

【請求項４】　ソースプログラムを入力して解析する第１のステップと、解析された結果に基づいて命令コードを生成する第２のステップと、生成された命令コードをオブジェクト・モジュール・ファイルに出力する第３のステップとを有する言語処理方法において、

前記ソースプログラムを解析する第１のステップと命令コードを生成する第２のステップとの間に、

オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておく第４のステップと、

オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、登録されている出力コードの各パターンを使用した回数を測定する第５のステップと、

前記コードパターンを登録する第４のステップによる登録内容および前記コードサイズを測定する第５のステップによる測定結果を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換える第６のステップとを含むことを特徴とする言語処理方法。

【請求項５】　前記コードパターンを登録する第４のステップが、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録するステップを含むことを特徴とする請求項４に記載の言語処理方法。

【請求項６】　前記コードパターン選択ステップが、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定するステップを含むことを特徴とする請求項４または請求項５に記載の言語処理方法。

【発明の詳細な説明】

【０００１】

【発明が属する技術分野】本発明は、言語処理装置および言語処理方法に関し、特にプログラム作成者が指定したコードサイズの範囲内で実行速度の速いオブジェクトを生成することができる言語処理装置及び言語処理方法に関する。

【０００２】

【従来の技術】言語処理プログラムが入力したソースプログラムを翻訳し、その結果として生成したファイルをオブジェクト・モジュール・ファイル（以下、オブジェクトと呼ぶ）と言う。

【０００３】マイクロコンピュータ、特にシングルチップマイクロコンピュータにおいては、コストの低減を図るため、性能面で制限が生じる傾向にある。従って、言語処理プログラムにおいては、言語処理を行った結果であるオブジェクトの実行速度（以下、単に実行速度と呼ぶ）が速いこと、及びオブジェクトのサイズ（以下、コードサイズと呼ぶ）が小さいことが望まれている。

【０００４】従来の言語処理装置では、入力されたソースプログラムに対して、コードサイズと実行速度のどちらかを重視した最適化を行ったオブジェクトの出力を行っている。

【０００５】図１０に従来の言語処理装置の構成を示す。

【０００６】図１０において、言語処理装置８２は、入力したソース・プログラム・ファイル８１を構文解析部

８３で解析する。構文解析の終了後、コード生成部８４に制御を移し、解析された結果およびプログラム作成者の指定に基づいて命令を選択して出力コードを生成する。コード生成の終了後、オブジェクト出力部８５に制御を移し、オブジェクト・モジュール・ファイル８６を出力する。

【０００７】コード生成部８４において、言語処理装置８２が複数の出力コードパターンを生成し、いずれかの出力コードパターンを選択できる場合には、プログラム作成者の選択により出力コードパターンの選択を行う。すなわち、コードサイズを優先する出力を指定した場合には、コードサイズ優先コード生成部８４１に制御を移し、コードサイズが小さくなるようにコード生成を行う。コードサイズ優先コード生成部８４１では、実行速度の向上よりもコードサイズの縮小を優先するため、生成されたオブジェクトの実行速度が遅くなることがあり得る。一方、プログラム作成者が実行速度を優先する出力を指定した場合には、実行速度優先コード生成部８４２に制御を移し、実行速度が速くなるようにコード生成を行う。実行速度優先コード生成部８４２では、コードサイズの縮小よりも実行速度の向上を優先するため、コードサイズが増大することがあり得る。

【０００８】次に、図１１を参照して従来の処理のアルゴリズムを説明する。

【０００９】従来の言語処理方式では、まず、プログラム作成者が入力したソースプログラム・ファイルの構文解析を行う（ステップ９０１）。次に、プログラム作成者がコードサイズを優先する指定をしたか、または実行速度を優先する指定をしたかを判断し（ステップ９０２）、コードサイズを優先する指定があった場合には、コード生成で複数の出力コードパターンを選択できるところで、コードサイズが小さくなるように出力コードを生成する（ステップ９０３）。一方、実行速度を優先する指定があった場合には、コード生成で複数の出力コードパターンを選択できるところで、実行速度が速くなるように出力コードを生成する（ステップ９０４）。そして、出力コードの生成が終了した後、オブジェクトを出力する（ステップ９０５）。

【００１０】このような従来技術の例として、例えば文献「マイクロソフトＣ　オプティマイジング　コンパイラ　ユーザーズガイド」（１９８８年マイクロソフト株式会社）に開示された技術がある。同文献には、言語処理装置実行時のオプションにより、オブジェクトのコードサイズを優先する出力を行うか、オブジェクトの実行速度を優先する出力を行うかを選択して出力コードを生成する技術が記載されている。

【００１１】

【発明が解決しようとする課題】上述した従来の言語処理技術では、プログラム作成者がオブジェクトに対してコードサイズの縮小と実行速度の向上とを同時に望んだ場合に、コードサイズと実行速度の両方に対してプログラム作成者が望む条件を十分に満たすオブジェクトを出力することが困難であるという問題点があった。その理由は、上記従来の技術では、コードサイズの縮小を優先するか、実行速度の向上を優先するかの二者択一の選択しかできず、コードサイズを優先する指定では実行速度が遅くなる場合があり、実行速度を優先する指定ではコードサイズが大きくなってしまう場合があるためである。コードサイズ及び実行速度は命令コードに依存するが、最小かつ最速の命令が常に存在するとは限らない。従って、コードサイズを最小にすると実行速度が遅くなることがあり、実行速度を最速にするとコードサイズが大きくなることがある。

【００１２】本発明の目的は、プログラム作成者が指定したコードサイズで、実行速度の速いオブジェクトを生成することができる言語処理装置および言語処理方法を提供することである。

【００１３】本発明の他の目的は、プログラム作成者自身がオブジェクトのコードサイズを自由に決定できる言語処理装置および言語処理方法を提供することである。

【００１４】

【課題を解決するための手段】上記の目的を達成するため、本発明の言語処理装置は、ソースプログラムを入力して解析する構文解析部と、解析された結果に基づいて命令コードを生成するコード生成部と、生成された命令コードをオブジェクト・モジュール・ファイルに出力するオブジェクト出力部とを備える言語処理装置において、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておくコードパターン登録部と、オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、前記コードパターン登録部に登録されている出力コードの各パターンを使用した回数を測定するコードサイズ測定部と、前記コードサイズ測定部の測定結果と前記コードパターン登録部の登録内容を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換えるコードパターン選択部とを備えることを特徴とする。

【００１５】請求項２の本発明の言語処理装置は、前記コードパターン登録部が、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録することを特徴と

する。

【００１６】請求項３の本発明の言語処理装置は、前記コードパターン選択部が、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定することを特徴とする。

【００１７】また、上記の目的を達成する本発明の言語処理方法は、ソースプログラムを入力して解析する第１のステップと、解析された結果に基づいて命令コードを生成する第２のステップと、生成された命令コードをオブジェクト・モジュール・ファイルに出力する第３のステップとを有する言語処理方法において、前記ソースプログラムを解析する第１のステップと命令コードを生成する第２のステップとの間に、オブジェクトのサイズを小さくすることを優先した場合の生成コードと、オブジェクトの実行速度を速くすることを優先した場合の生成コードとで異なるコードを出力するパターンについて、それぞれのコード生成方法、コード生成によるコードサイズの差を登録しておく第４のステップと、オブジェクトのサイズを小さくすることを優先してコード生成を行った場合のオブジェクトのサイズと、オブジェクトの実行速度を速くすることを優先してコード生成を行った場合のオブジェクトのサイズとを測定し、かつ生成された命令コード中で、登録されている出力コードの各パターンを使用した回数を測定する第５のステップと、前記コードパターンを登録する第４のステップによる登録内容および前記コードサイズを測定する第５のステップによる測定結果を参照して、プログラム作成者が指定したオブジェクトのコードサイズ内に収まるように、オブジェクトの実行速度を優先した生成コードの一部または全部を、オブジェクトのサイズを優先した生成コードに置き換える第６のステップとを含むことを特徴とする。

【００１８】請求項５の本発明の言語処理方法は、前記コードパターンを登録する第４のステップが、オブジェクトの実行速度の劣化を抑え、オブジェクトのサイズをより小さくすることができる順番にコードパターンを登録するステップを含むことを特徴とする。

【００１９】請求項６の本発明の言語処理方法は、前記コードパターン選択ステップが、各コードパターンの使用回数のうち、オブジェクトのサイズを優先するコード生成を行う回数を決定するステップを含むことを特徴とする。

【００２０】

【発明の実施の形態】以下、本発明の実施例について図面を参照して詳細に説明する。

【００２１】図１は、本発明の第１の実施例による言語処理装置の構成を示すブロック図である。

【００２２】図示のように、言語処理装置１２は、ソース・プログラム・ファイル１１を入力して内容を解析する構文解析部１３と、構文解析の結果を元にコード生成を行うコード生成部１７と、コード生成部１７で生成されたコードをオブジェクト・モジュール・ファイル１９として出力するオブジェクト出力部１８を備えると共に、コードサイズを優先する時と実行速度を優先する時とで出力コードが異なるコードパターンの情報が登録してあるコードパターン登録部１６と、コードサイズを測定しコードパターン登録部１６に登録してあるコードパターンの使用回数を測定するコードサイズ測定部１４と、出力コードをプログラム作成者の指定したコードサイズ以内にするため出力コードパターンを選択するコードパターン選択部１５とを備える。以上の構成において、構文解析部１３とコード生成部１７とオブジェクト出力部１８の機能は、従来のものと同様である。

【００２３】コードパターン登録部１６は、コードサイズを優先した場合と実行速度を優先した場合とでコード生成が異なる部分に関して、コードサイズを優先した場合のコードパターンと、実行速度を優先した場合のコードパターンと、コードサイズを優先した場合のコードパターンと実行速度を優先した場合のコードパターンとのコードサイズの差とを登録しておく。コードサイズを優先した場合と実行速度を優先した場合とでオブジェクトのコードサイズが異なる場合は、コードパターン登録部１６に登録されたコードパターンが少なくとも１つは存在し、登録されたコードパターンによって生成コードの違いが生じることになる。生成コードの違いが生じると、コードサイズを優先した場合と実行速度を優先した場合とのオブジェクトのコードサイズに差が生じ、一般に、実行速度を優先した場合のコードサイズが大きくなる。

【００２４】コードサイズ測定部１４は、コードサイズを優先した場合のオブジェクトのコードサイズ、および実行速度を優先した場合のオブジェクトのコードサイズを測定する。コードサイズの測定は、コード生成部１７と同様の処理方法で実現可能である。測定した２つのコードサイズ値はコードパターン選択部１５の処理において使用される。

【００２５】コードサイズ測定部１４によるコードサイズ測定処理と並行して、コードパターン登録部１６に登録されたコードパターンが、それぞれ何回使用されたかを測定し、コードパターン登録部１６に各コードパターンの使用回数を保存する。コードパターン登録部１６に保存された各コードパターンの使用回数は、コードパターン選択部１５の処理において使用される。

【００２６】コードパターン選択部１５は、コードパターン登録部１６に保存された各コードパターンに対して、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うかを選択するための情報を設定する。設定した情報を参照することによって、オブジェクトのコードサイズがプログラム作成者の指定したコードサイズ以内に収まるように、実行速度を優先した場合のオブジェクトの生成コードの一部または全部

を、コードサイズを優先した場合の生成コードに置き換えることができる。

【００２７】コード生成部１７は、コードパターン選択部１５による選択結果にしたがって、コードサイズ優先コード生成部１７１または実行速度優先コード生成部１７２においてコード生成を行う。コードサイズ優先コード生成部１７１では、コードサイズが小さくなるようにコード生成を行い、実行速度優先コード生成部１７２では、実行速度が速くなるようにコード生成を行う。コード生成部１７でのコード生成は、コードサイズを優先したコード生成と、実行速度を優先したコード生成が混在することになる。コード生成の終了後は、オブジェクト出力部１８に制御を移し、オブジェクトを出力する。

【００２８】次に、図２及び図３のフローチャートを参照して本実施例の処理の詳細を説明する。

【００２９】まず、プログラム作成者が希望するコードサイズを指定する（図２、ステップ２０１）。指定した値は、メモリ上の特定された領域（「ｕｓｅｒ」と称す）に格納される。

【００３０】プログラム作成者が希望するコードサイズを指定すると、構文解析部１３が、ソース・プログラム・ファイル１１の内容を構文解析する（ステップ２０２）。構文解析が終了すると、コードサイズ測定部１４が、コードサイズを優先した場合のオブジェクトのコードサイズを測定する（ステップ２０３）。ステップ２０３のコードサイズ測定処理と並行して、コードパターン登録部１６に登録されているコードパターンの使用回数を測定する（ステップ２０４）。

【００３１】図４はコードパターン登録部１６の登録内容の例である。コードパターン登録部１６には、コードサイズを優先した場合の生成コード１Ａ、２Ａ、・・・、実行速度を優先した場合の生成コード１Ｂ、２Ｂ、・・・、コードサイズを優先した場合の生成コードと実行速度を優先した場合の生成コードとのコードサイズ差ｄ［１］、ｄ［２］、・・・、及びコードパターンの使用回数ｃｎｔ［１］、ｃｎｔ［２］、・・・がそれぞれメモリ上の領域として確保されている。ｋ番目のコードパターンについては、コードサイズを優先した場合の生成コードｋＡ、実行速度を優先した場合の生成コードｋＢ、ｋＡとｋＢとのコードサイズ差ｄ［ｋ］が登録されている。ｋＡとｋＢとは生成コードは違うが、同じ動作をするコードである。ステップ２０４で測定した使用回数は、領域ｃｎｔ［ｋ］に設定される。

【００３２】ステップ２０３のコードサイズ測定により得られたコードサイズは、メモリ上の領域（「ｃｏｄｅ」と称す）に格納される（ステップ２０５）。

【００３３】ステップ２０５が終了すると、コードサイズ測定部１４が、構文解析部１１での構文解析の結果を再度参照し、実行速度を優先した場合のオブジェクトのコードサイズを測定する（ステップ２０６）。ステップ２０６のコードサイズ測定により得られたコードサイズは、メモリ上の領域（「ｓｐｅｅｄ」と称す）に格納される（ステップ２０７）。

【００３４】ステップ２０７の終了後は、コードパターン選択部１５に制御を移す。まず、出力予定のオブジェクトのコードサイズの初期設定を行い、メモリ上の領域（「ｓｉｚｅ」と称す）に格納する（ステップ２０８）。初期設定値は、実行速度を優先した場合のコードサイズ「ｓｐｅｅｄ」とする。

【００３５】ステップ２０８の終了後は、コードパターン登録部１６に登録されている各コードパターンについて、メモリ上の領域ｆｌａｇ［１］、ｆｌａｇ［２］、・・・を「０」に初期化する（ステップ２０９）。ｆｌａｇ［１］、ｆｌａｇ［２］、・・・とは、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うかを判断するためのフラグを表す。フラグｆｌａｇ［ｋ］は、コードパターン登録部１６のｋ番目に登録されているコードパターンについてのフラグを表す。フラグｆｌａｇ［］は、「０」のときにフラグが降りた状態を示し、「０」以外のときにフラグが立った状態を示すものとする。コード生成では、それぞれのコードパターンに対し、フラグが立っているならばコードサイズを優先するコード生成を行い、フラグが降りているならば実行速度を優先するコード生成を行うことになる。ステップ２０９のフラグｆｌａｇ［］の初期化後は、フラグｆｌａｇ［］はすべて降りた状態になるので、このままコード生成を行ったと仮定すると、複数のコード生成が可能なすべての記述において、実行速度を優先したコード生成を行うことになる。

【００３６】ステップ２０９のフラグｆｌａｇ［］の初期化が終了すると、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」とコードサイズを優先した場合のコードサイズ「ｃｏｄｅ」とを比較する（図３、ステップ２１０）。プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、コードサイズを優先した場合のコードサイズ「ｃｏｄｅ」以下である場合は、コードサイズを優先するコード生成を行うフラグｆｌａｇ［１］、ｆｌａｇ［２］、・・・をすべて立て（ステップ２１１）、コード生成部１７に制御を移し、ステップ２１７以降のコード生成処理を行う。

【００３７】ステップ２１０において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、コードサイズを優先した場合のコードサイズ「ｃｏｄｅ」よりも大きい場合は、コードパターン登録部１６に登録されているコードパターンの登録番号を表すメモリ上の領域「ｉ」を「１」に初期化し（ステップ２１２）、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」と出力予定のオブジェクトのコードサイズ［ｕｓｅｒ］を比較する（ステップ２１３）。プログラム作成者が指定したコードサイズ「ｕｓｅｒ」、出力予定のオブジェクトの

コードサイズ［ｕｓｅｒ］よりも小さい場合は、フラグｆｌａｇ［ｉ］を立て、コードパターン登録部１６に登録されているｉ番目のコードパターンについて、すべての使用箇所でコードサイズ優先のコード生成を行うことにする（ステップ２１４）。

【００３８】ステップ２１４において、一部のコード生成が実行速度を優先するコード生成からコードサイズを優先するコード生成に代わることにより、出力予定のオブジェクトのコードサイズは、コードパターン登録部１６のｉ番目に登録されているコードサイズ差ｄ［ｉ］と使用回数ｃｎｔ［ｉ］との積の分だけ小さくなるので、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」からｄ［ｉ］とｃｎｔ［ｉ］の積を減算した値を、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」として再設定する（ステップ２１５）。

【００３９】ステップ２１５における出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」の再設定が終了すると、コードパターン登録部１６のコードパターンの登録番号「ｉ」の内容に「１」を加え（ステップ２１６）、ステップ２１３に戻って、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」と再設定した出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」とを比較する。

【００４０】ステップ２１３において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」以上である場合には、コード生成部１７に制御を移し、ステップ２１７以降のコード生成処理を行う。

【００４１】コード生成部１７では、コード生成が終了したか否かを判断する（ステップ２１７）。コード生成が終了していない場合は、複数の生成コードが存在するか否かを判断する（ステップ２１８）。これは従来の処理と同じである。生成コードが１つしかない場合は、従来の方法と同様にコード生成を行う（ステップ２１９）。複数の生成コードが存在する場合は、コードパターン登録部１６の何番目に登録されたコードパターンであるかを判断する（ステップ２２０）。ステップ２２０で求めたコードパターンの登録番号を「ｋ」とする。

【００４２】この後、コードパターン登録部１６のｋ番目に登録されているコードパターンに対して、フラグｆｌａｇ［ｋ］が立っているか否かを判断する（ステップ２２１）。フラグｆｌａｇ［ｋ］が立っていれば、コードサイズ優先コード生成部１７１に制御を移し、コードサイズを優先するコードパターンでコード生成を行う（ステップ２２２）。また、フラグｆｌａｇ［ｋ］が降りていれば、実行速度優先コード生成部１７２に制御を移し、実行速度を優先するコードパターンでコード生成を行う（ステップ２２３）。ステップ２１９、ステップ２２２、またはステップ２２３においてコード生成を行った後は、ステップ２１７に戻り、コード生成が終了したか否かを判断する。コード生成が終了していなければ次のコード生成を行う。すべてのコード生成が終了したならば、オブジェクト出力部１８に制御を移し、オブジェクト・モジュール・ファイル１９を出力する。

【００４３】以上の処理を、具体的な例を挙げて説明する。図５は、コードパターン登録部１６の登録内容の一例である。図示のように、この例では、コードサイズの単位をｂｙｔｅ、実行速度の単位をｃｌｏｃｋとし、６パターンのコードパターンが登録されていたものとする。コードパターンの使用回数ｃｎｔ［１］、ｃｎｔ［２］、・・・は、ステップ２０４の処理において設定される。

【００４４】所定のソース・プログラム・ファイルについて、コードサイズ測定部１４におけるコードサイズの測定により、プログラム作成者が指定したオブジェクトのコードサイズ「ｕｓｅｒ」が１５００ｂｙｔｅ、コードサイズを優先した場合のオブジェクトのコードサイズ「ｃｏｄｅ」が１２００ｂｙｔｅ、実行速度を優先した場合のオブジェクトのコードサイズ「ｓｐｅｅｄ」が１６００ｂｙｔｅであるものとする。また、コードパターン登録部１６に登録されているコードパターンの使用回数ｃｎｔ［］が、登録番号順に２２回、２６回、３４回、５２回、４０回、６８回であるものとする。

【００４５】以上の条件で、ステップ２０８において、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」は、実行速度を優先した場合のオブジェクトのコードサイズ「ｓｐｅｅｄ」と同じ１６００ｂｙｔｅに設定される。

【００４６】ステップ２０９において、複数の生成コードが存在する場合にコードサイズを優先するコード生成を行うかどうかを判断するためのフラグｆｌａｇ［］のすべてを「０」に初期化する。

【００４７】ステップ２１０におけるコードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｃｏｄｅ」（１２００ｂｙｔｅ）との比較で、コードサイズ「ｕｓｅｒ」はコードサイズ「ｃｏｄｅ」より大きいので、ステップ２１２に処理を移してコードパターン登録部１６のコードパターンの登録番号を表すメモリ上の領域「ｉ」を「１」に初期化する。

【００４８】ステップ２１３におけるコードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｓｉｚｅ」（１６００ｂｙｔｅ）との比較において、コードサイズ「ｕｓｅｒ」はコードサイズ「ｓｉｚｅ」より小さいので、ステップ２１４で、フラグｆｌａｇ［１］を立てる。図５から、コードサイズ差ｄ［１］は１ｂｙｔｅ、コードパターン使用回数ｃｎｔ［１］は２２回なので、ステップ２１５において、コードサイズ「ｓｉｚｅ」は（１＊２２）ｂｙｔｅ減って１６７８ｂｙｔｅとなり、ステップ２１６において、コードパターンの登録番号「ｉ」は「２」となる。

【００４９】ステップ２１６の処理が終了すると、ステ

ップ２１３に戻って、再びコードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｓｉｚｅ」（１５７８ｂｙｔｅ）とを比較する。コードサイズ「ｓｉｚｅ」はコードサイズ「ｕｓｅｒ」より大きいので、ステップ２１４においてフラグｆｌａｇ［2］を立てる。図５から、コードサイズ差ｄ［2］は２ｂｙｔｅ、コードパターン使用回数ｃｎｔ［2］は２６回なので、ステップ２１５において、コードサイズ「ｓｉｚｅ」は（２＊２６）ｂｙｔｅ減って１６２６ｂｙｔｅとなり、ステップ２１６において、コードパターンの登録番号「ｉ」は「3」となる。

【００５０】ステップ２１３に戻って、コードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｓｉｚｅ」（１５２６ｂｙｔｅ）とを比較すると、コードサイズ「ｓｉｚｅ」はコードサイズ「ｕｓｅｒ」より大きいので、ステップ２１４において、フラグｆｌａｇ［3］を立てる。図５から、コードサイズ差ｄ［3］は１ｂｙｔｅ、コードパターン使用回数ｃｎｔ［3］は３４回なので、ステップ２１５において、コードサイズ「ｓｉｚｅ」は（１＊３４）ｂｙｔｅ減って１４９２ｂｙｔｅとなり、ステップ２１６において、コードパターンの登録番号「ｉ」は「4」となる。

【００５１】ステップ２１３に戻って、コードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｓｉｚｅ」（１４９２ｂｙｔｅ）とを比較すると、コードサイズ「ｓｉｚｅ」はコードサイズ「ｕｓｅｒ」以下となるので、ステップ２１７以降のコード生成に処理を移す。

【００５２】コード生成においては、コードパターン登録部１６に登録されているコードパターンについて、登録番号が１～３のコードパターンはフラグｆｌａｇ［］が立っているので、コードサイズを優先するコード生成を行い、登録番号が４～６のコードパターンはフラグｆｌａｇ［］が降りているので、実行速度を優先するコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクトを出力する。

【００５３】出力されたオブジェクトは、コードサイズが１４９２ｂｙｔｅとなり、プログラム作成者の希望するオブジェクトのコードサイズ１５００ｂｙｔｅ以下という条件を満たしている。また、実行速度を優先するコードパターンが含まれているために、コードサイズを優先した場合のオブジェクトに比して実行速度が速くなる。

【００５４】次に、本発明の第２の実施例について図面を参照して詳細に説明する。

【００５５】上記の第１の実施例では、コードパターン登録部１６に登録されている各コードパターンについて、フラグｆｌａｇ［］が立っているか否かによって、コードサイズを優先するコード生成か実行速度を優先するコード生成かを決定し、決定後はそのコードパターンを使用する箇所では常に同じコード生成を行っていた。また、第１の実施例で登録されているコードパターンには、登録順番についての規則を設けていないため、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換える順番に関する効率は考慮していない。本実施例では、実行速度の劣化をより抑えたコード生成を行うために、２つの改善を行っている。第１の改善は、登録されている各コードパターンに対して、生成コードの置き換えに関する優先順位を持たせることである。第２の改善は、登録されている各コードパターンに対して、コードサイズを優先するコード生成と実行速度を優先するコード生成とを混在させることによって、出力予定のオブジェクトのサイズをプログラム作成者によって指定されたコードサイズ以内にするための、生成コードの置き換えを最小限に留めることである。

【００５６】本実施例の言語処理装置の構成は、コードパターン登録部１６に追加登録する項目があること以外は、図１に示した第１の実施例の言語処理装置の構成と同様であるため、全体の構成の説明は省略する。

【００５７】図８は、本実施例のコードパターン登録部１６の登録内容である。コードパターン登録部１６には、図４に示した第１の実施例の登録内容に加え、実行速度を優先した場合の生成コードとコードサイズを優先した場合の生成コードとの実行速度差ｃ［1］、ｃ［2］、・・・がメモリ上の領域として確保され、登録されている。ｃ［k］はコードパターン登録部１６のｋ番目に登録された生成コードｋＡと生成コードｋＢとの実行速度差を表す。

【００５８】第１の実施例では、コードパターン登録部１６に登録されているコードパターンの登録順番には規則がなかったが、本実施例では登録順番に優先順位を付け、優先順位の高いコードパターンから順番に登録しておく。優先順位の高いコードパターンとは、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合に、実行速度の劣化が少なく、かつコードサイズの縮小に効果的なコードパターンである。コードパターン登録部１６に登録されたコードサイズ差ｄ［］から実行速度差ｃ［］を減算した値ｄ［］－ｃ［］を優先基準とし、ｄ［］－ｃ［］の値が大きいコードパターンを優先し、値が同じものについては実行速度差ｃ［］が小さいコードパターンを優先する。優先順位を付けることにより、コードパターン選択部１５は、実行速度を優先するコード生成からコードサイズを優先するコード生成への置き換えを、実行速度の劣化が少ないコードパターンから順番に行うことが可能となる。

【００５９】また、本実施例ではコードパターン選択部１５の処理が前記だい１の実施例の場合と異なる。第１の実施例では、フラグｆｌａｇ［1］、ｆｌａｇ［2］、・・・によって、コードサイズを優先するコード生成を行うか実行速度を優先するコード生成を行うか

を判断していた。本実施例では、コードパターン登録部１６に登録されている各コードパターンに対して、コードサイズを優先するコード生成を行う回数を設定し、設定した回数だけコードサイズを優先するコード生成を行い、残りの回数は実行速度を優先するコード生成を行う。

【００６０】以下、図６及び図７のフローチャートを参照して本実施例の処理の詳細を説明する。

【００６１】まず、図２に図示された、コードパターン選択部１６によるステップ２０８までの処理を行う。処理内容は第１の実施例と同様であるため説明を省略する。

【００６２】ステップ２０８までの処理が終了すると、コードパターン登録部１６に登録されている各コードパターンについて、コードサイズを優先するコードパターンの使用回数を表すメモリ上の領域ｕｓｅ［1］、ｕｓｅ［2］、・・・を「0」に初期化する（図７、ステップ５０１）。ｕｓｅ［k］は、コードパターン登録部１６のｋ番目に登録されているコードパターンについての、コードサイズを優先するコードパターンの使用回数を表す。

【００６３】ステップ５０１の初期化が終了すると、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」とコードサイズを優先した場合のコードサイズ「ｃｏｄｅ」とを比較する（ステップ５０２）。プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、コードサイズを優先した場合のコードサイズ「ｃｏｄｅ」以下である場合は、コードサイズを優先するコードパターンの使用回数ｕｓｅ［］に、コードパターンの使用回数ｃｎｔ［］の値を設定する（ステップ５０３）。コードパターン登録部１６のｋ番目に登録されているコードパターンに関しては、コードサイズを優先するコードパターンの使用回数ｕｓｅ［k］に、コードパターンの使用回数ｃｎｔ［k］の値が設定される。ステップ５０３の設定により、すべてのコードパターンにおいてコードサイズを優先するコード生成を行うことになる。ステップ５０３の処理が終了すると、コード生成部１７に制御を移し、ステップ５１１以降のコード生成処理を行う。

【００６４】ステップ５０２において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、コードサイズを優先した場合のコードサイズ「ｃｏｄｅ」よりも大きい場合は、コードパターン登録部１６に登録されているコードパターンの登録番号を表すメモリ上の領域「ｉ」を「１」に初期化し（ステップ５０４）、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」と出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」を比較する（ステップ５０５）。プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」以上である場合は、コード生成部１７に制御を移し、ステップ５１１以降のコード生成処理を行う。

【００６５】ステップ５０５において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」より小さい場合は、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」から、コードパターン登録部１６のｉ番目に登録されているコードサイズ差ｄ［ｉ］と使用回数ｃｎｔ「ｉ」との積を減算した値と、プログラム作成者が指定したオブジェクトのコードサイズ「ｕｓｅｒ」とを比較する（ステップ５０６）。ステップ５０６において比較に用いられる値ｓｉｚｅ－ｄ［ｉ］＊ｃｎｔ「ｉ」は、コードパターン登録部１６のｉ番目のコードパターンを使用しているすべての箇所で、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合の出力予定のオブジェクトのコードサイズである。

【００６６】ステップ５０６において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、ｓｉｚｅ－ｄ［ｉ］＊ｃｎｔ［ｉ］以下である場合は、コードパターン登録部１０のｉ番目に登録されているコードパターンに関して、コードサイズを優先するコードパターンの使用回数ｕｓｅ［ｉ］にコードパターンの使用回数ｃｎｔ［ｉ］の値を設定し（ステップ５０７）、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」に、ｓｉｚｅ－ｄ［ｉ］＊ｃｎｔ［ｉ］を再設定する（ステップ５０８）。ステップ５０８における出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」の再設定が終了すると、コードパターン登録部１６のコードパターンの登録番号「ｉ」の内容に「１」加え（ステップ５０９）、ステップ５０５に戻って、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」と再設定した出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」とを比較する。

【００６７】ステップ５０６において、プログラム作成者が指定したコードサイズ「ｕｓｅｒ」が、ｓｉｚｅ－ｄ［ｉ］＊ｃｎｔ［ｉ］より大きい場合は、コードパターン登録部１６のｉ番目に登録されているコードパターンに関して、出力予定のオブジェクトのコードサイズ「ｕｓｅｒ」以下になるような、コードサイズを優先するコードパターンの使用回数を計算し、ｕｓｅ［ｉ］に設定する（ステップ５１０）。ステップ５１０でｕｓｅ［ｉ］に設定する値は、（ｕｓｅｒ－ｓｉｚｅ）／ｄ［ｉ］以上の最小の整数値である。ステップ５１０におけるｕｓｅ［ｉ］の設定が終了すると、コード生成部１７に制御を移し、ステップ５１１以降のコード生成処理を行う。

【００６８】コード生成部１７では、コード生成が終了したか否かを判断する（図７、ステップ５１１）。コード生成が終了していない場合は、複数の生成コー・・・亨存在するか否かを判断する（ステップ５１２）。これは従来の処理と同じである。生成コードが１つしかない

場合は、従来の方法と同様にコード生成を行う（ステップ５１３）。複数の生成コードが存在する場合は、コードパターン登録部１６の何番目に登録されたコードパターンであるかを判断する（ステップ５１４）。ステップ５１４で求めたコードパターンの登録番号をｋとする。

【００６９】コードパターン登録部１６のｋ番目に登録されているコードパターンに対して、コードサイズを優先するコードパターンの使用回数ｕｓｅ［ｋ］が「０」か否かを調べ（ステップ５１５）、ｕｓｅ［ｋ］が「０」であるならば、実行速度優先コード生成部１７２に制御を移し、実行速度を優先するコードパターンでコード生成を行う（ステップ５１６）。ｕｓｅ［ｋ］が「０」でなければ、コードサイズ優先コード生成部１７１に制御を移し、コードサイズを優先するコードパターンでコード生成を行い（ステップ５１７）、ｕｓｅ［ｋ］の内容から「１」を引く（ステップ５１８）。ステップ５１３、ステップ５１６、またはステップ５１７、５１８においてコード生成を行った後は、ステップ５１１に戻り、コード生成が終了したか否かを判断し、終了していなければ次のコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクト出力部１８に制御を移し、オブジェクト・モジュール・ファイル１９を出力する。

【００７０】以上の処理を、具体的な例を挙げて説明する。図９は、コードパターン登録部１６の登録内容の一例である。図示のように、この例では、コードサイズの単位をｂｙｔｅ、実行速度の単位をｃｌｏｃｋとする。図９に登録されているコードパターンは、図５と同じものであるが、実行速度を優先するコード生成からコードサイズを優先するコード生成に置き換えた場合に、実行速度の劣化が少なく、かつコードサイズの縮小に効果的なコードパターンから順に登録されている。優先基準としては、コードサイズ差ｄ［］から実行速度差ｃ［］を引いた値が大きいコードパターンを優先し、値が同じものについては、実行速度差ｃ［］が小さいコードパターンを優先する。コードパターンの使用回数ｃｎｔ［１］、ｃｎｔ［２］、・・・については、ステップ２０４において設定される。

【００７１】所定のソース・プログラム・ファイルについて、コードサイズ測定部１４におけるコードサイズの測定により、プログラム作成者が指定したオブジェクトのコードサイズ「ｕｓｅｒ」が１５００ｂｙｔｅ、コードサイズを優先した場合のオブジェクトのコードサイズ「ｃｏｄｅ」が１２００ｂｙｔｅ、実行速度を優先した場合のオブジェクトのコードサイズ「ｓｐｅｅｄ」が１６００ｂｙｔｅであるものとする。また、コードパターン登録部１６に登録されているコードパターンの使用回数ｃｎｔ［］が、登録番号順に５２回、６８回、４０回、２２回、２６回、３４回であるものとする。

【００７２】ステップ２０８において、出力予定のオブジェクトのコードサイズ「ｓｉｚｅ」は、実行速度を優先した場合のオブジェクトのコードサイズｓｐｅｅｄと同じ１６００ｂｙｔｅに設定される。

【００７３】ステップ５０１において、コードサイズを優先するコードパターンの使用回数ｕｓｅ［］のすべてを「０」に初期化する。

【００７４】ステップ５０２におけるコードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｃｏｄｅ」（１２００ｂｙｔｅ）との比較で、コードサイズ「ｕｓｅｒ」はコードサイズ「ｃｏｄｅ」より大きいので、ステップ５０４に処理を移してコードパターン登録部１６のコードパターンの登録番号を表すメモリ上の領域「ｉ」を「１」に初期化する。

【００７５】ステップ５０５におけるコードサイズ「ｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズ「ｓｉｚｅ」（１６００ｂｙｔｅ）との比較で、コードサイズ「ｕｓｅｒ」はコードサイズ「ｓｉｚｅ」より小さいので、ステップ５０６に処理を移す。図９から、コードサイズ差ｄ［１］は２ｂｙｔｅ、コードパターン使用回数ｃｎｔ［１］は５２回なので、ステップ５０６において、コードサイズｕｓｅｒ」（１５００ｂｙｔｅ）とコードサイズｓｉｚｅ－ｄ［１］＊ｃｎｔ［１］（（１６００－２＊５２）ｂｙｔｅ）を比較する。

【００７６】コードサイズ「ｕｓｅｒ」はコードサイズｓｉｚｅ－ｄ［１］＊ｃｎｔ［１］より大きいので、ステップ５１０に処理を移して、ｕｓｅ［１］に（ｕｓｅｒ－ｓｉｚｅ）／ｄ［１］以上の最小の整数値を設定する。（１６００－１５００）／２＝５０であることから、ｕｓｅ［１］には５０を設定する。即ち、１番目に登録されているコードパターンの使用回数５２回のうち、５０回はコードサイズを優先するコード生成を行い、残りの２回は実行速度を優先するコード生成を行うことになる。

【００７７】１番目のコードパターンについてのコード生成を行う場合は、ｕｓｅ［１］の値が「０」になるまで、ステップ５１７に処理を移して、コードサイズを優先するコード生成を行い、ステップ５１８においてｕｓｅ［１］の内容から「１」を引く。ｕｓｅ［１］の値が「０」になったならば、ステップ５１６に処理を移して、実行速度を優先するコード生成を行う。２番目以降のコードパターンについては、ｕｓｅ［］の値が「０」であるので、ステップ５１６に処理を移して、実行速度を優先するコード生成を行う。そして、すべてのコード生成が終了した後、オブジェクトを出力する。

【００７８】出力されたオブジェクトはコードサイズが１５００ｂｙｔｅとなり、プログラム作成者の希望するオブジェクトのコードサイズ１５００ｂｙｔｅ以下という条件を満たしている。また、実行速度を優先したコードパターンが含まれているために、コードサイズを優先した場合のオブジェクトよりも実行速度が速くなる。さ

らに、実行速度の劣化を抑えた生成コードの置き換えを行っているので、第１の実施例よりも実行速度が速くなっている。

【００７９】以上のように、本発明では、コードサイズを優先した場合のコードサイズおよび登録したコードパターンの使用回数、実行速度を優先した場合のコードサイズを測定し、実行速度を優先した場合の生成コードの一部または全部を、プログラム作成者が指定したコードサイズ以内に収まるまで、コードサイズを優先した場合の生成コードに置き換えてオブジェクトを出力する。このため、従来のコードサイズを優先した場合の生成コードと、実行速度を優先した場合の生成コードとの間の任意のサイズのコードを生成することができ、結果として、プログラム作成者が指定したコードサイズ以内で、実行速度の速いオブジェクトを出力することができる。

【００８０】以上好ましい実施例をあげて本発明を説明したが、本発明は必ずしも上記実施例に限定されるものではない。

【００８１】

【発明の効果】以上説明したように、本発明の言語処理装置及び言語処理方法によれば、実行速度を優先するコード生成の一部を、コードサイズを優先するコード生成に置き換えることにより、実行速度の劣化を抑えると共にオブジェクトのコードサイズの調整ができるため、オブジェクトの実行速度をあまり損なわずに、コードサイズの短縮ができるという効果がある。

【００８２】これにより、実行速度を優先するコード生成を行ったオブジェクトが、プログラム作成者の希望するコードサイズに収まらなかった場合にプログラム作成者自身がオブジェクトを書き換えてオブジェクトのコードサイズを調整するといったプログラム作成者の手間を省くことができるという効果がある。

【００８３】また、オブジェクトのコードサイズを調節して、実行速度を向上することができるため、コードサイズを優先するコード生成を行ったオブジェクトが、プログラム作成者の希望するコードサイズよりずっと小さくなった場合に、オブジェクトのコードサイズをプログラム作成者の希望するコードサイズまで増やすことによって、より実行速度の速いオブジェクトを作成することができるという効果がある。

【図面の簡単な説明】

【図１】　本発明の第１の実施例による言語処理装置の構成を示すブロック図である。

【図２】　第１の実施例によるコードサイズ測定部からコード生成部までの動作を示すフローチャートである。

【図３】　第１の実施例によるコードサイズ測定部からコード生成部までの動作を示すフローチャートである。

【図４】　第１の実施例による言語処理装置のコードパターン登録部の登録内容を示す図である。

【図５】　第１の実施例による言語処理装置のコードパターン登録部の登録内容の具体例を示す図である。

【図６】　本発明の第２の実施例によるコードパターン選択部からコード生成部までの動作を示すフローチャートである。

【図７】　本発明の第２の実施例によるコードパターン選択部からコード生成部までの動作を示すフローチャートである。

【図８】　第２の実施例による言語処理装置のコードパターン登録部の登録内容を示す図である。

【図９】　第２の実施例による言語処理装置のコードパターン登録部の登録内容の具体例を示す図である。

【図１０】　従来の言語処理装置の構成を示すブロック図である。

【図１１】　従来の言語処理装置の動作を示すフローチャートである。

【符号の説明】

１１　ソースプログラムファイル
１２　言語処理装置
１３　構文解析部
１４　コードサイズ測定部
１５　コードパターン選択部
１６　コードパターン登録部
１７　コード生成部
１８　オブジェクト出力部
１９　オブジェクトモジュールファイル
１７１　コードサイズ優先コード生成部
１７２　実行速度優先コード生成部

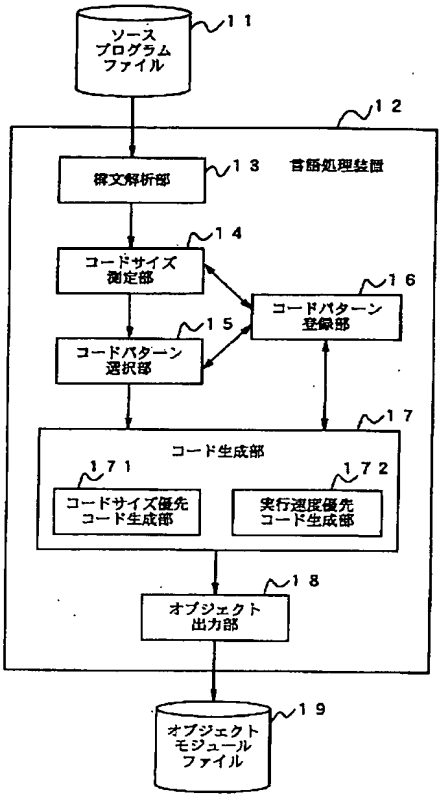
【図４】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	パターン使用回数
1	1A	1B	d[1]	cnt[1]
2	2A	2B	d[2]	cnt[2]
⋮	⋮	⋮	⋮	⋮
n	nA	nB	d[n]	cnt[n]

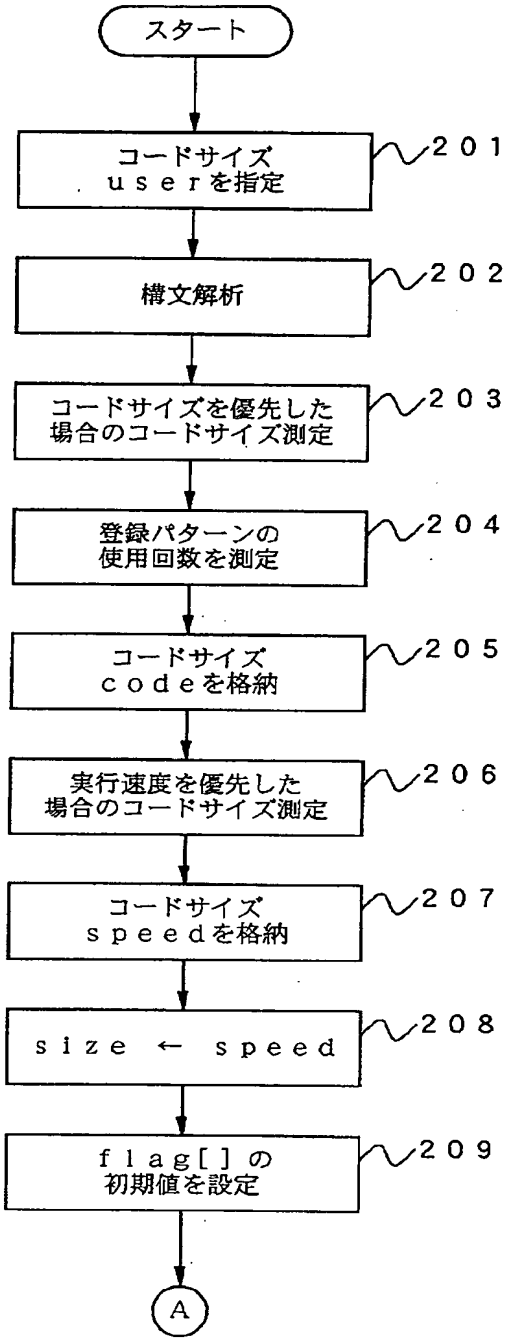
【図５】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	パターン使用回数
1	1A (2byte, 6clock)	1B (8byte, 4clock)	1byte	22
2	2A (4byte, 9clock)	2B (6byte, 6clock)	2byte	26
3	3A (3byte, 9clock)	3B (4byte, 5clock)	1byte	34
4	4A (5byte, 8clock)	4B (7byte, 7clock)	2byte	52
5	5A (2byte, 8clock)	5B (5byte, 5clock)	3byte	40
6	6A (3byte, 7clock)	6B (4byte, 6clock)	1byte	68

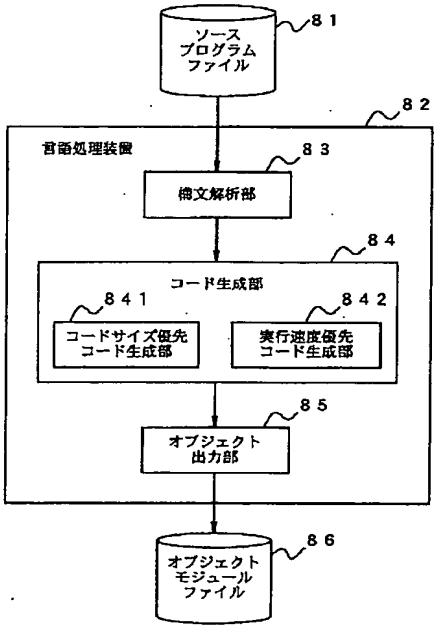
【図１】


ソース
プログラム
ファイル
11
12
構文解析部
13
言語処理装置
コードサイズ
測定部
14
コードパターン
登録部
16
コードパターン
選択部
15
17
コード生成部
171
172
コードサイズ優先
コード生成部
実行速度優先
コード生成部
18
オブジェクト
出力部
オブジェクト
モジュール
ファイル
19

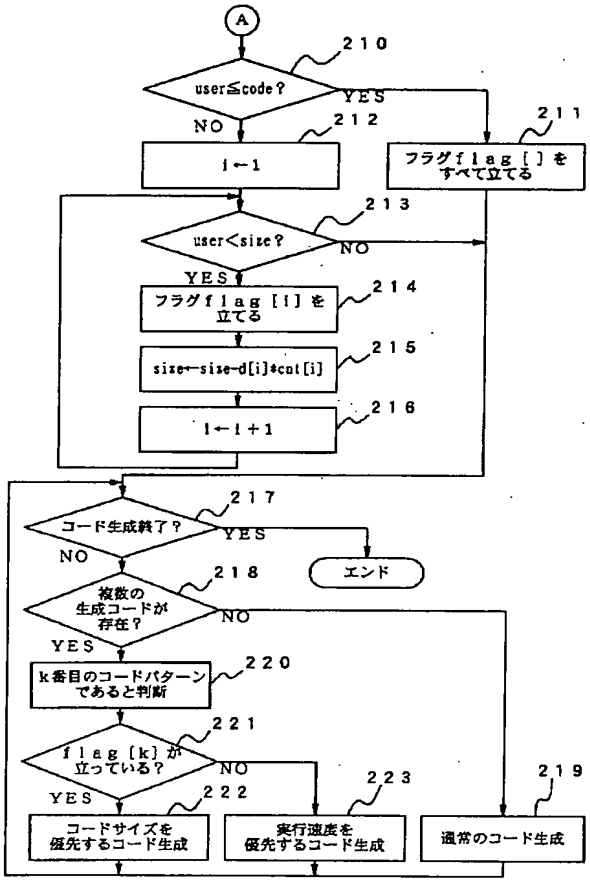
【図２】


スタート
コードサイズ
ｕｓｅｒを指定
201
構文解析
202
コードサイズを優先した
場合のコードサイズ測定
203
登録パターンの
使用回数を測定
204
コードサイズ
ｃｏｄｅを格納
205
実行速度を優先した
場合のコードサイズ測定
206
コードサイズ
ｓｐｅｅｄを格納
207
ｓｉｚｅ　←　ｓｐｅｅｄ
208
ｆｌａｇ[ ] の
初期値を設定
209
A

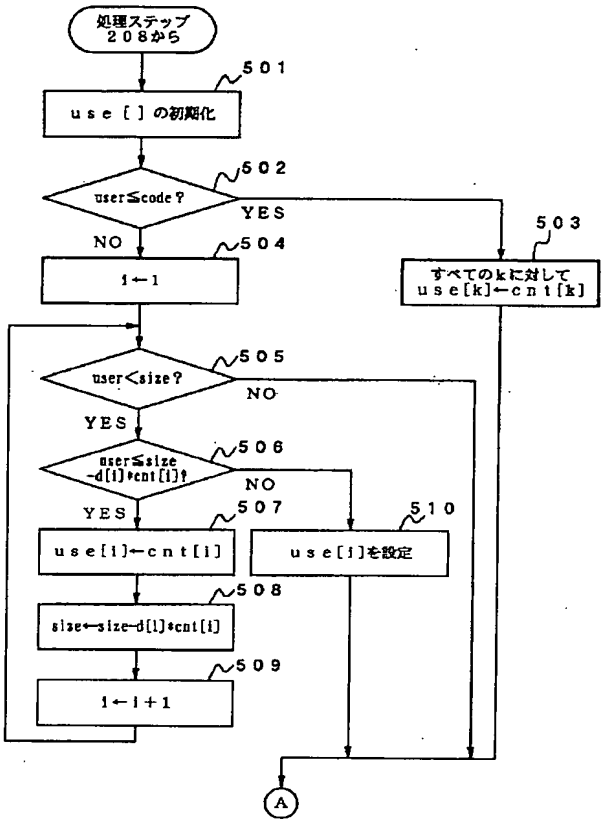
【図１０】


ソース
プログラム
ファイル
81
82
言語処理装置
83
構文解析部
84
コード生成部
841
842
コードサイズ優先
コード生成部
実行速度優先
コード生成部
85
オブジェクト
出力部
オブジェクト
モジュール
ファイル
86

【図３】



【図６】



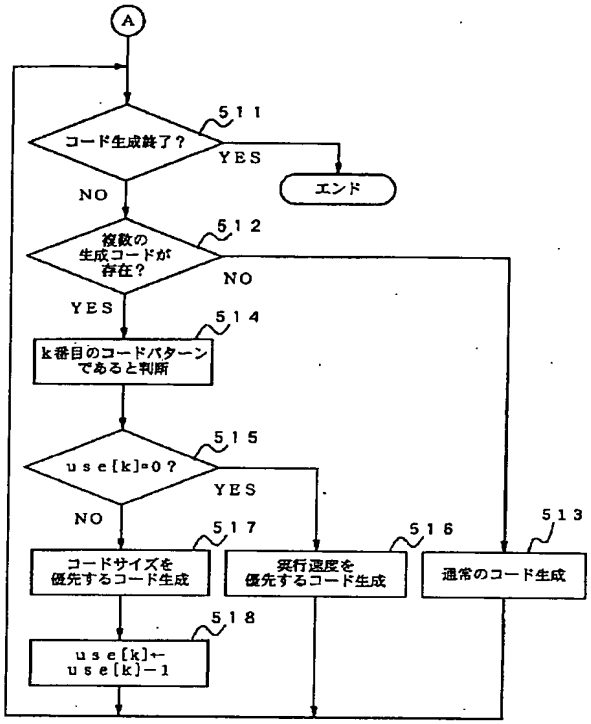
【図８】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	実行速度差	パターン使用回数
1	1A	1B	d[1]	c[1]	cnt[1]
2	2A	2B	d[2]	c[2]	cnt[2]
⋮	⋮	⋮	⋮	⋮	⋮
n	nA	nB	d[n]	c[n]	cnt[n]

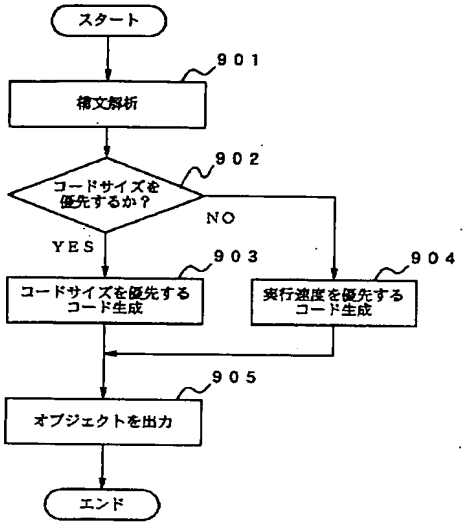
【図９】

登録番号	コードサインを優先するコードパターン	実行速度を優先するコードパターン	コードサイズ差	実行速度差	パターン使用回数
1	1A (5byte, 8clock)	1B (7byte, 7clock)	2byte	1clock	52
2	2A (3byte, 7clock)	2B (4byte, 6clock)	1byte	1clock	68
3	3A (2byte, 8clock)	3B (5byte, 5clock)	3byte	3clock	40
4	4A (2byte, 6clock)	4B (3byte, 4clock)	1byte	2clock	22
5	5A (4byte, 9clock)	5B (6byte, 6clock)	2byte	3clock	26
6	6A (3byte, 8clock)	6B (4byte, 5clock)	1byte	4clock	34

【図７】


A
５１１
コード生成終了？
YES
エンド
NO
５１２
複数の
生成コードが
存在？
NO
YES
５１４
k番目のコードパターン
であると判断
５１５
use[k]=0？
YES
NO
５１７
コードサイズを
優先するコード生成
５１６
実行速度を
優先するコード生成
５１３
通常のコード生成
５１８
use[k]←
use[k]−1

【図１１】


スタート
９０１
構文解析
９０２
コードサイズを
優先するか？
NO
YES
９０３
コードサイズを優先する
コード生成
９０４
実行速度を優先する
コード生成
９０５
オブジェクトを出力
エンド